Բովանդակություն

[Ներածություն 4](#_Toc166517048)

[1.Խնդրի դրվածքը 5](#_Toc166517049)

[2․Տվյալների բազայի ստեղծում և մշակում 6](#_Toc166517050)

[2․1․ DDL հրամանների կիրառում 6](#_Toc166517051)

[Տվյալների բազայի և աղյուսակների ստեղծումը 6](#_Toc166517052)

[Աղյուսակների դաշտերի սահմանափակումները 9](#_Toc166517053)

[Աղյուսակների միջև կապերի որոշում 10](#_Toc166517054)

[Դիագրամներ 10](#_Toc166517055)

[2․2․ DML հրամանների կիրառում 11](#_Toc166517056)

[DML հրամանների կիրառմամաբ աղյուսակների լրացում 11](#_Toc166517057)

[2.3. Պահպանող մոդուլներ 12](#_Toc166517058)

[Պրոցեդուրաներ 12](#_Toc166517059)

[Ֆունկցիաներ 15](#_Toc166517060)

[Տրիգերներ 17](#_Toc166517061)

[2․4․ Ստեղծել view-եր և դրանց միջոցով կազմակերպել DML հարցումներ 19](#_Toc166517062)

[Եզրակացություն 22](#_Toc166517063)

[Գրականություն 23](#_Toc166517064)

# Ներածություն

Տվյալների բազայի կառավարման համակարգը (ՏԲԿՀ) ծրագրային հավելված է կամ հավելվածների հավաքածու, որը հեշտացնում է կառուցվածքային և արդյունավետ կերպով տվյալների ստեղծումը, պահպանումը, կազմակերպումը և առբերումը: Այն օգտատերերին տրամադրում է գործիքներ՝ սահմանելու, կառավարելու, մանիպուլյացիայի ենթարկելու և տվյալների բազաներում պահվող տվյալների հարցումներ ստանալու համար՝ ապահովելով տվյալների ամբողջականությունը, անվտանգությունը և միաժամանակյա հսկողությունը:

SQL (Structured Query Language) ստանդարտացված հայցերի ծրագրավորման լեզու է, որը հատուկ նախագծված է ռելացիոն տվյալների բազաները կառավարելու և շահագործելու համար: SQL-ը թույլ է տալիս օգտատերերին կատարել գործողությունների լայն շրջանակ, ներառյալ տվյալների բազաներից տվյալների առբերումը (SELECT), նոր տվյալների ներմուծումը (INSERT), առկա տվյալների խմբագրումը (UPDATE) և տվյալների ջնջումը (DELETE): Բացի այդ, SQL-ն աջակցում է տվյալների բազայի սխեմաների ստեղծման և փոփոխման, սահմանափակումների սահմանման և օգտատերերի մուտքի թույլտվությունների կառավարման գործողություններին: SQL-ն լայնորեն օգտագործվում է տարբեր ոլորտներում և հավելվածներում, ներառյալ, բայց չսահմանափակվելով վեբ մշակման, տվյալների վերլուծության, բիզնեսի հետախուզության և ծրագրային ապահովման մշակման մեջ՝ շնորհիվ իր պարզության, ճկունության և տվյալների բազայի հարթակներում լայն տարածման: Ավելին, SQL-ը հնարավորություն է տալիս կատարել բարդ հարցումներ, միացումներ և ագրեգացիաներ՝ դարձնելով այն հզոր գործիք տվյալների մեծ հավաքածուներից արժեքավոր պատկերացումներ քաղելու համար:

# 1.Խնդրի դրվածքը

Մշակել տվյալների հենք

Թեմա Ծաղկի սրահ

Աշխատանքի մոտավոր նկարագրություն

Տեղեկատվություն ծաղիկների դասակարգման, ծառայությունների մատուցման, ապրանքի առաքման, աշխատակիցների և մատակարարների մասին։

Ֆինանսական և վիճակագրական հաշվետվություններ սրահի գործունեության ֆինանսական և վիճակագրական, դուրս գրված ապրանքների, յուրաքանչյուր վաճառողի գործունեության մասին։

Աշխատանքի պահանջները

1․Ստեղծել անհրաժեշտ բազան և աղյուսակները։

2․ Աղյուսակների դաշտերին տալ համապատասխան սահմանափակումներ։

3․ Կապել այդ աղյուսակները։

4․ Աղյուսակները լրացնել տվյալներով։

5․ Կատարել DDL, DML հարցումներ։

6․ Կիրառել պայմանի օպերատորներ, case-եր և ցիկլեր հարցումներում:

7․ Ստեղծել պահպանող մոդուլներ ինչպիսիք են՝ պրոցոդուրա, ֆունկցիա, տրիգեր։

8․ Ստեղծել view-եր և դրանց միջոցով կազմակերպել DML հարցումներ։

# 2․Տվյալների բազայի ստեղծում և մշակում

# 2․1․ DDL հրամանների կիրառում

## Տվյալների բազայի և աղյուսակների ստեղծումը

Ծաղկի սրահ տվյալների բազայի կառուցման համար օգտագործվել է 6 աղյուսակ, որոնց հիման վրա կատարվել է հետագա տվյալների մշակումը։ Այդ աղյուսակների կառուցվածքը, դաշտերի նկարագրությունը և SQL server management studio ծրագրի միջավայրում տվյալների բազայի և աղյուսակների ստեղծման համար անհրաժեշտ հրամանները հետևյալն են․

CREATE DATABASE Caxki\_srah;

go

use Caxki\_srah;

go

--Pashton

CREATE TABLE Position (

ID INT NOT NULL PRIMARY KEY, -- Ամբողջ տիպի պարտադիր լրացվող դաշտ առաջնային բանալի

position\_name NVARCHAR(50) NOT NULL, -- Տեքստային տիպի պարտադիր լրացվող եզակի դաշտ

salary MONEY NOT NULL --Դրամային տիպ պարտադիր լրացվող դաշտ );

Position աղյուսակը ներկայացնում է սրահի պաշտոնները և ներառում է հետևյալ 3 դաշտերը․

ID – Պաշտոնի կոդ։

position\_name – Պաշտոնի անվանում։

salary – Պաշտոնի համար սահմանված աշխատավարձ։

--Ashxatoxner

CREATE TABLE Employee (

id INT NOT NULL PRIMARY KEY, --Ամբողջ տիպի պարտադիր լրացվող դաշտ առաջնային բանալի

first\_name NVARCHAR(50) NOT NULL, --Տեքստային տիպի պարտադիր լրացվող դաշտ

last\_name NVARCHAR(50) NOT NULL, --Տեքստային տիպի պարտադիր լրացվող դաշտ

phone\_number NVARCHAR(20) NOT NULL, --Տեքստային տիպի պարտադիր լրացվող դաշտ

address\_ NVARCHAR(50) NOT NULL, --Տեքստային տիպի պարտադիր լրացվող դաշտ

birthday DATE NOT NULL, --Ամսաթվային տիպ պարտադիր լրացվող դաշտ

SSN NVARCHAR(20) NOT NULL, --Տեքստային տիպի պարտադիր լրացվող եզակի դաշտ

email NVARCHAR(50) NOT NULL, --Տեքստային տիպի պարտադիր լրացվող եզակի դաշտ

gender NVARCHAR(20) NOT NULL, --Տեքստային տիպի պարտադիր լրացվող դաշտ

position\_id INT NOT NULL ); -- Ամբողջ տիպի պարտադիր լրացվող դաշտ արտաքին բանալի

Employee աղյուսակում ներկայացված են սրահի աշխատակիցները իրենց անձնական տվյալներով։ Աղյուսակը ներառում է հետևյալ 10 դաշտերը․

id – Աշխատողի կոդ։

first\_name – Աշխատողի անվան դաշտ ։

last\_name - Աշխատողի ազգանվան դաշտ։

phone\_number – Աշխատողների հեռախոսահամարի դաշտ։

address\_ - Աշխատողի բնակության հասցեի դաշտ։

birthday – Աշխատողի ծննդյան տվյալների դաշտ։

SSN – Աշխատողի սոցիալական համարանիշի դաշտ։

email – Աշխատողի էլեկտրոնային հասցեի դաշտ։

gender – Աշխատողի սեռի դաշտ։

position\_id – Աշխատողի պաշտոնի կոդ։

--Vacharq

CREATE TABLE Sales (

Sale\_id INT NOT NULL PRIMARY KEY, --Ամբողջ տիպի պարտադիր լրացվող դաշտ առաջնային բանալի

flower\_id INT NOT NULL, --Ամբողջ տիպի պարտադիր լրացվող դաշտ արտաքին բանալի

sales\_quantity INT NOT NULL, --Ամբողջ տիպի պարտադիր լրացվող դաշտ

price MONEY NOT NULL, --Դրամային տիպ պարտադիր լրացվող դաշտ

sale\_date DATE NOT NULL, --Ամսաթվային տիպ պարտադիր լրացվող, լռելյայն GETDATE() արժեքով դաշտ

employee\_id INT NOT NULL, --Ամբողջ տիպի պարտադիր լրացվող դաշտ արտաքին բանալի

shipping\_id INT, --Ամբողջ տիպի դաշտ արտաքին բանալի

payment\_method NVARCHAR(50) NOT NULL --Տեքստային տիպի պարտադիր լրացվող, լռելյայն 'Cash' արժեքով դաշտ);

Sales աղյուսակում ներկայացված են սրահի վաճառքի հետ կապված տվյալները։ Աղյուսակը ներառում է հետևյալ 8 դաշտերը․

Sale\_id – Վաճառքի կոդ։

sales\_quantity – Վաճառված ծաղիկների քանակ։

price – Վաճառքի արժեք։

sale\_date – Վաճառքի ամսաթիվ։

payment\_method – Վաճառք կատարելու եղանակ։

flower\_id – Վաճառված ծաղիկների կոդ։

employee\_id – Վաճառք իրականացնող աշխատողների կոդ։

shipping\_id – Վաճառված ծաղիկների առաքման կոդ։

--Caxikner

CREATE TABLE Flowers (

F\_id INT NOT NULL PRIMARY KEY, --Ամբողջ տիպի պարտադիր լրացվող դաշտ առաջնային բանալի

flower\_name NVARCHAR(50) NOT NULL, --Ամբողջ տիպի պարտադիր լրացվող եզակի դաշտ

flower\_quantity INT, --Ամբողջ տիպի պարտադիր լրացվող դաշտ

flower\_price MONEY NOT NULL, --Դրամային տիպ պարտադիր լրացվող դաշտ

supplier\_id INT NOT NULL --Ամբողջ տիպի պարտադիր լրացվող դաշտ արտաքին բանալի);

Flowers աղյուսակում ներկայացված են սրահի ծաղիկների հետ կապված տվյալները։ Աղյուսակը ներառում է հետևյալ 5 դաշտերը․

F\_id - Ծաղկի կոդ։

flower\_name – Ծաղկի անվանում։

flower\_quantity – Ծաղիկների քանակ։

flower\_price – Ծաղկի արժեք։

supplier\_id – Ծաղկի մատակարարի կոդ։

--Araqum

CREATE TABLE Shipping (

Ship\_id INT NOT NULL PRIMARY KEY, --Ամբողջ տիպի պարտադիր լրացվող դաշտ առաջնային բանալի

shipping\_date DATE NOT NULL, --Ամսաթվային տիպ պարտադիր լրացվող դաշտ

shipping\_address NVARCHAR(255) NOT NULL, --Ամբողջ տիպի պարտադիր լրացվող դաշտ

shipping\_pay MONEY, --Դրամային տիպ պարտադիր լրացվող դաշտ);

Shipping աղյուսակում ներկայացված են սրահի ծաղիկների աքումների հետ կապված տվյալները։ Աղյուսակը ներառում է հետևյալ 4 դաշտերը․

Ship\_id – Առաքման կոդ։

shipping\_date – Առաքման ամսաթիվ։

shipping\_address – Առաքման հասցե։

shipping\_pay – Առաքման վճար։

--Matakarar

CREATE TABLE Supplier (

Sup\_id INT NOT NULL PRIMARY KEY, --Ամբողջ տիպի պարտադիր լրացվող դաշտ առաջնային բանալի

firstname NVARCHAR(50) NOT NULL, --Ամբողջ տիպի պարտադիր լրացվող դաշտ

lastname NVARCHAR(50) NOT NULL, --Ամբողջ տիպի պարտադիր լրացվող դաշտ

sup\_phone NVARCHAR(20) NOT NULL, --Ամբողջ տիպի պարտադիր լրացվող դաշտ

sup\_email NVARCHAR(50) NOT NULL, --Ամբողջ տիպի պարտադիր լրացվող դաշտ

s\_address NVARCHAR(50) NOT NULL, --Ամբողջ տիպի պարտադիր լրացվող դաշտ);

Supplier աղյուսակում ներկայացված են սրահի առաքիչների հետ կապված տվյալները։ Աղյուսակը ներառում է հետևյալ 6 դաշտերը․

Sup\_id – Մատակարարի եզակի կոդ։

firstname – Մատակարարի անվան դաշտ։

lastname - Մատակարարի ազգանվան դաշտ։

sup\_phone - Մատակարարի հեռախոսահամարի դաշտ։

sup\_email - Մատակարարի էլեկտրոնային հասցե։

sup\_address – Մատակարարի բնակության հասցե։

## Աղյուսակների դաշտերի սահմանափակումները

Սահմանափակումները օգտագործվում են սահմանափակելու տվյալների տեսակը, որը կարող է մուտք գործել աղյուսակ։ Դրանց շնորհիվ ապահովվում է տվյալների ճշգրտությունն ու ամբողջականությունը։

Primary key և not null սահմանափակումները աղյուսակների ստեղծման ժամանակ իրականացվել է, իսկ foreign key սահմանափակումը alter հրամանի միջոցով ներկայացված է հաջորդիվ։ Բազայում ստեղծված մյուս սահմանափակումներից որոշները իրականացվել են create իսկ որոշները alter հրամանի միջոցով։ Դրանք սահմանվել են հետևյալ կերպ․

--Checks

ALTER TABLE Employee --Աշխատողների սեռը կարող է լինել արական կամ իգական

ADD CONSTRAINT CK\_Gender CHECK (gender IN ('Male', 'Female'));

ALTER TABLE Employee --Աշխատողները պետք է լինեն չափահաս

ADD CONSTRAINT CK\_BD CHECK (DATEDIFF(YEAR, birthday, GETDATE()) >= 18);

ALTER TABLE Employee --Սոցիալական քարտի նիշերի երկարությունը պետք է հավասար լինի 10-ի

ADD CONSTRAINT CK\_SSN CHECK (SSN NOT LIKE '%[^0-9]%' AND LEN(SSN) = 10);

ALTER TABLE Employee --Մեյլը պետք է պարունակի (@) սիմվոլը որին հաջորդում է (.) սիմվոլը

ADD CONSTRAINT CK\_mail CHECK (email LIKE '%@%.%');

ALTER TABLE Employee --Հեռախոսահամարը պետք է սկսվի +374 կոդով և պարունակի 12 նիշ

ADD CONSTRAINT CK\_Phone CHECK (phone\_number NOT LIKE '+374%[^0-9]%' AND LEN(phone\_number) = 12);

ALTER TABLE Supplier

ADD CONSTRAINT CK\_SPhone CHECK (sup\_phone NOT LIKE '+374%[^0-9]%' AND LEN(sup\_phone) = 12);

ALTER TABLE Supplier

ADD CONSTRAINT CK\_Semail CHECK (sup\_email LIKE '%@%.%')

ALTER TABLE Sales –Վճարման եղանակը կարող է լինել կանխիկ կամ քարտային

ADD CONSTRAINT CK\_PaymentMethod CHECK (payment\_method IN ('Cash', 'Card'));

--UNIQUE

--1

ALTER TABLE Employee

ADD UNIQUE(SSN); --Սոցիալական քարտի համարը պետք է լինի եզակի

--2

ALTER TABLE Employee

ADD CONSTRAINT UC\_Email UNIQUE (email);--Մեյլը պետք է լինի եզակի

--3

ALTER TABLE Position

ADD CONSTRAINT UC\_PositionName UNIQUE (position\_name);--Պաշտոնի անվանումը պետք է չկրկնվի

--4

ALTER TABLE Flowers

ADD CONSTRAINT UC\_FlowersName UNIQUE (flower\_name);--Ծաղիկների անուններըպետք է չկրկնվեն

--DEFAULTS

--1

ALTER TABLE Sales --Վճարման եղանակի համար լռելյայն սահմանվել է կանխիկ տարբերակը

ADD CONSTRAINT DF\_PaymentMethod DEFAULT 'Cash' FOR payment\_method;

--2

ALTER TABLE Sales --Վաճառքի ամսաթիվը լռելյայն սահմանվել է GETDATE() ժամանակահատվածը

ADD CONSTRAINT DF\_SaleDate DEFAULT GETDATE() FOR sale\_date;

--3

ALTER TABLE Flowers --Ծաղիկների սկզբնական քանակը լռելյայն սահմանվել է 0

ADD CONSTRAINT DF\_FlowerQuantity DEFAULT 0 FOR flower\_quantity;

## Աղյուսակների միջև կապերի որոշում

Աղյուսակների ստեղծումից հետո դրանք անհրաժեշտ է կապել։ Իմ աղյուսակների մեջ գոյություն ունի 1- և կապերը։

Աղյուսակները իրար կապելու համար օգտագործվել է alter հրամանը։ Այդ գործողղության կոդը հետևյալն է․

--1

ALTER TABLE Employee

ADD CONSTRAINT FK\_Employee

FOREIGN KEY (position\_id) REFERENCES Position(ID);

--2

ALTER TABLE Sales

ADD CONSTRAINT FK\_Sales

FOREIGN KEY (flower\_id) REFERENCES Flowers(F\_id);

--3

ALTER TABLE Sales

ADD CONSTRAINT FK\_Saless

FOREIGN KEY (employee\_id) REFERENCES Employee(id);

--4

ALTER TABLE Sales

ADD CONSTRAINT FK\_sale

FOREIGN KEY (shipping\_id) REFERENCES Shipping(Ship\_id);

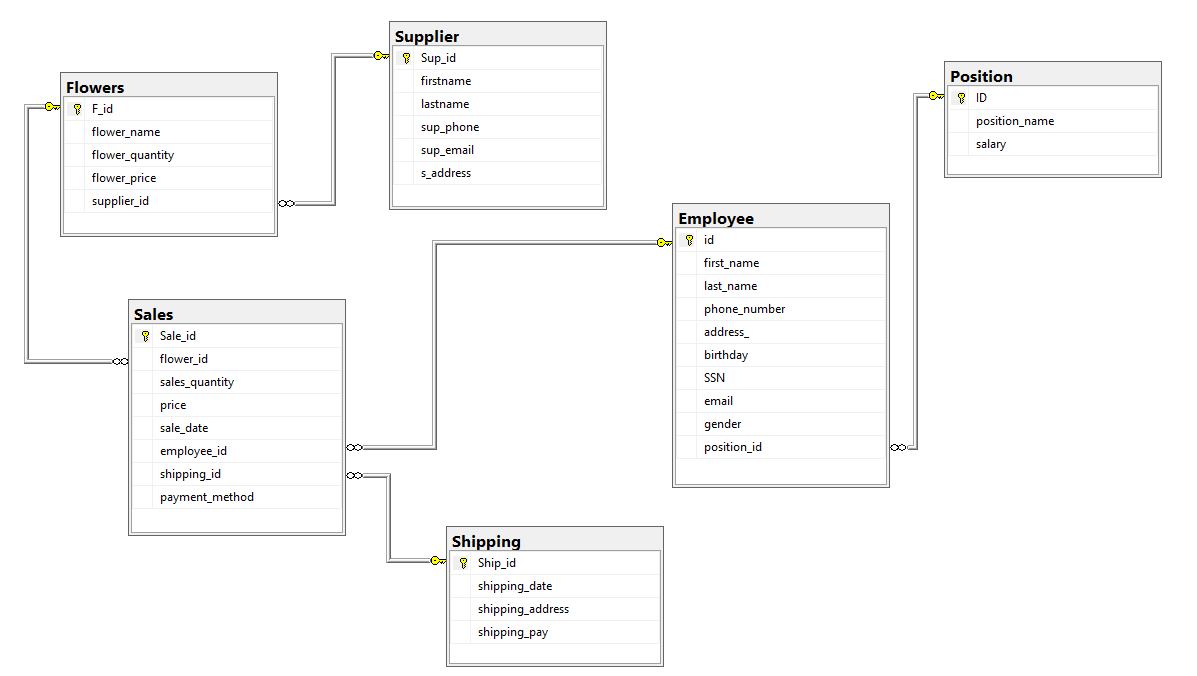
--5

ALTER TABLE Flowers

ADD CONSTRAINT FK\_Flowers

FOREIGN KEY (supplier\_id) REFERENCES Supplier(Sup\_id);

## Դիագրամներ



# 2․2․ DML հրամանների կիրառում

## DML հրամանների կիրառմամաբ աղյուսակների լրացում

Աղյուսակները լրացնելու համար օգտագործվում է insert հրամանը։ Այն կատարվում է հետևյալ կերպ․

INSERT INTO Position(ID, position\_name, salary)

VALUES(3,'Havaqarar', 80000);



INSERT INTO Employee(id, first\_name, last\_name, phone\_number, address\_, birthday, SSN, email, gender, position\_id)

VALUES

(1, 'Garik', 'Sargsyan', '+37499658741', 'Yerevan, Armenian St 123', '1990-01-01', '2514631548', 'g.sargsyan@gmail.com', 'Male', 5);



INSERT INTO Supplier(Sup\_id, firstname, lastname, sup\_phone, sup\_email, s\_address)

Values

(1, 'Armen', 'Adunc', '+37499254111', 'armen.adunc@mail.ru', 'Yerevan, Isakov St 64');



INSERT INTO Flowers(F\_id, flower\_name, flower\_quantity, flower\_price, supplier\_id)

VALUES

(1, 'Vard', 100, 800, 1);



INSERT INTO Shipping(Ship\_id, shipping\_date, shipping\_address, shipping\_pay)

VALUES

(1, '2024-02-24', 'Yerevan, Tumanyan St', 0);



INSERT INTO Sales(Sale\_id, flower\_id, sales\_quantity, price, sale\_date, employee\_id, shipping\_id, payment\_method)

VALUES

(11, 1, 50, 1500, '2024-02-24', 7, 1, 'Card');



# 2.3. Պահպանող մոդուլներ

## Պրոցեդուրաներ

1․Ստեղծել պրոցեդուրա, որը հաշվում է կոնկրետ ծաղկի վաճառքից ստացված եկամուտը։

CREATE PROCEDURE CalculateSalesRevenueForFlower

@flowerName NVARCHAR(50), @salesRevenue MONEY OUTPUT

AS BEGIN

SET NOCOUNT ON;

IF EXISTS ( SELECT 1

FROM Sales s

JOIN Flowers f ON s.flower\_id = f.F\_id

WHERE f.flower\_name = @flowerName )

BEGIN SELECT @salesRevenue = SUM(s.sales\_quantity \* s.price)

FROM Sales s

JOIN Flowers f ON s.flower\_id = f.F\_id

WHERE f.flower\_name = @flowerName;

END ELSE

BEGIN SET @salesRevenue = 0;

END

END;

DECLARE @salesRevenue MONEY;

EXEC CalculateSalesRevenueForFlower 'Pion', @salesRevenue OUTPUT;

SELECT @salesRevenue AS SalesRevenueForVard;



2․ Պրոցեդուրան տրամադրում է վաճառքի հաշվետվություն տվյալ ժամանակահատվածի համար։

CREATE PROCEDURE SalesReport

@StartDate DATE, @EndDate DATE

AS BEGIN

DECLARE @SalesReport TABLE (

ProductName NVARCHAR(50),

TotalSales INT,

TotalQuantitySold INT,

TotalRevenue MONEY,

TotalCost MONEY,

TotalProfit MONEY );

INSERT INTO @SalesReport (ProductName, TotalSales, TotalQuantitySold, TotalRevenue, TotalCost, TotalProfit)

SELECT f.flower\_name AS ProductName,

f.flower\_name AS ProductName,

COUNT(\*) AS TotalSales,

SUM(s.sales\_quantity) AS TotalQuantitySold,

SUM(s.sales\_quantity \* s.price) AS TotalRevenue,

SUM(f.flower\_price\*f.flower\_quantity) AS TotalCost,

SUM(s.sales\_quantity \* s.price) - SUM(f.flower\_price\*f.flower\_quantity) AS TotalProfit

FROM Sales s

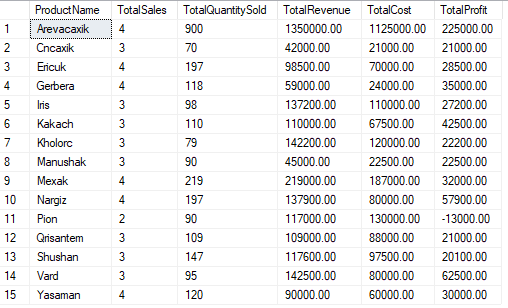
INNER JOIN Flowers f ON s.flower\_id = f.F\_id

WHERE s.sale\_date BETWEEN @StartDate AND @EndDate

GROUP BY f.flower\_name, s.flower\_id;

SELECT \* FROM @SalesReport; END;

EXEC SalesReport '2024-01-01', '2024-04-07';



3․Պրոցեդուրան սահմանում է բոնուս աշխատակիցների համար ովքեր վաճառել են որոշակի քանակով ծաղիկներ կամ բերել են որոշակի չափի եկամուտ։

CREATE PROCEDURE EmployeeBonuses

@MinFlowersSold INT, @MinTotalRevenue MONEY, @Bonus FLOAT

AS BEGIN

UPDATE Position

SET salary = salary + ISNULL(B.BonusAmount, 0)

FROM ( SELECT

e.position\_id,

SUM(s.sales\_quantity) AS TotalFlowersSold,

SUM(s.sales\_quantity \* (s.price - f.flower\_price)) AS TotalSalesRevenue,

CASE

WHEN SUM(s.sales\_quantity) > @MinFlowersSold OR SUM(s.sales\_quantity \* s.price ) > @MinTotalRevenue THEN SUM(s.sales\_quantity \* f.flower\_price) \* @Bonus

WHEN SUM(s.sales\_quantity) < @MinFlowersSold AND SUM(s.sales\_quantity \* s.price) < @MinTotalRevenue THEN 0

ELSE NULL

END AS BonusAmount

FROM Employee e

JOIN Sales s ON e.id = s.employee\_id

JOIN Flowers f ON s.flower\_id = f.F\_id

GROUP BY e.position\_id )

WHERE Position.ID = B.position\_id;

SELECT e.id AS EmployeeID,

e.first\_name + ' ' + e.last\_name AS EmployeeName,

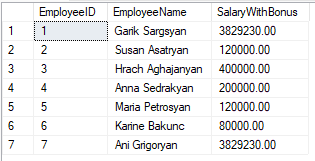
p.salary AS SalaryWithBonus

FROM Employee e

JOIN Position p ON e.position\_id = p.ID;

END;

EXEC EmployeeBonuses @MinFlowersSold = 300, @MinTotalRevenue = 10000, @Bonus = 0.05;



4․Ստեղծել պրոցեդուրա, որը վաճառքի գինը կնվազեցնի 5%-ով քանի դեռ ամբողջ վաճառքի գումարային արժեքը չի գերազանցում որոշակի սահմանը։ Եթե մինիմալ արժեքը փոքր է որոշակի սահմանից ընդհատել ցիկլը։

CREATE PROCEDURE UpdateSales

AS BEGIN

DECLARE @TotalSales MONEY, @MinAllowedValue MONEY = 0;

SELECT @TotalSales = SUM(price \* sales\_quantity)

FROM Sales;

WHILE (@TotalSales > 800000 AND @MinAllowedValue <= 5000)

BEGIN

BEGIN

UPDATE Sales

SET price = price \* 0.95;

END

SELECT @TotalSales = SUM(price \* sales\_quantity)

FROM Sales;

SELECT @MinAllowedValue = MIN(price \* sales\_quantity)

FROM Sales;

IF @MinAllowedValue <= 100000

BEGIN

PRINT 'Minimum value is less than or equal to 5000. Stopping the cycle.';

BREAK;

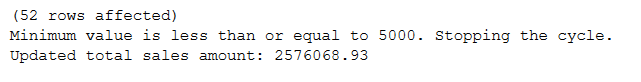
END

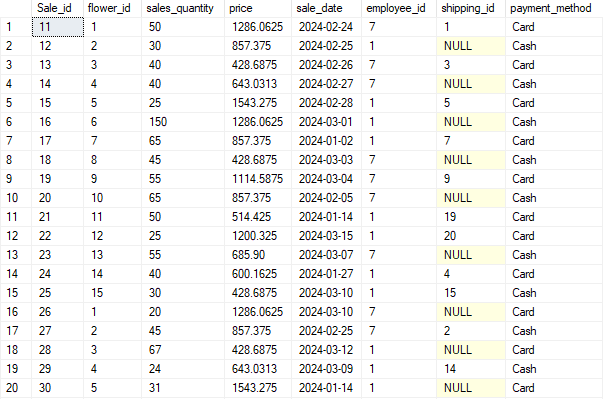
END;

PRINT 'Updated total sales amount: ' + CONVERT(VARCHAR, @TotalSales);

END;

EXEC UpdateSales;





## Ֆունկցիաներ

1․Ստեղծել ֆունկցիա, որը ստանում է վերջին մեկ ամսվա ընթացքում կատարված առաքումների միջին վճարը եթե այն գերազանցում է 1000 դրամը, հակառակ դեպքում ստանում է առաքումների քանակը։

CREATE FUNCTION GetShippingInfoForLastMonth ()

RETURNS NVARCHAR(100)

AS BEGIN

DECLARE @result NVARCHAR(100), @avg INT;

SELECT @avg = AVG(shipping\_pay)

FROM Shipping

WHERE MONTH(shipping\_date) = MONTH(DATEADD(MONTH, -1, GETDATE()));

IF @avg > 1000

BEGIN

SELECT @result = CONVERT(NVARCHAR(100), SUM(shipping\_pay))

FROM Shipping

WHERE MONTH(shipping\_date) = MONTH(DATEADD(MONTH, -1, GETDATE()));

END ELSE

BEGIN

SELECT @result = CONVERT(NVARCHAR(100), COUNT(\*))

FROM Shipping

WHERE MONTH(shipping\_date) = MONTH(DATEADD(MONTH, -1, GETDATE()));

END

RETURN @result;

END;

DECLARE @result NVARCHAR(100);

SET @result = dbo.GetShippingInfoForLastMonth();

SELECT @result AS ShippingInfoForLastMonth;



2․Ֆունկցիայի միջոցով փոխանցված պաշտոնի համաձայն ստանալ աշխատակիցների մասին տեղեկատվություն։

CREATE FUNCTION dbo.GetEmpByPosition (

@PositionName NVARCHAR(50) )

RETURNS @EmployeeTable TABLE (

EmployeeID INT,

EmployeeName NVARCHAR(100),

PhoneNumber NVARCHAR(20),

Email NVARCHAR(50),

Position NVARCHAR(50) )

AS BEGIN

DECLARE @EmployeePosition NVARCHAR(50);

SET @EmployeePosition = '';

WHILE @EmployeePosition != @PositionName

BEGIN SELECT TOP 1 @EmployeePosition = p.position\_name

FROM Employee e

INNER JOIN Position p ON e.position\_id = p.ID

WHERE p.position\_name > @EmployeePosition

ORDER BY p.position\_name;

IF @EmployeePosition IS NULL

BREAK;

IF @EmployeePosition = @PositionName

BEGIN

INSERT INTO @EmployeeTable (EmployeeID, EmployeeName, PhoneNumber, Email, Position)

SELECT e.id AS EmployeeID,

e.first\_name + ' ' + e.last\_name AS EmployeeName,

e.phone\_number AS PhoneNumber,

e.email AS Email,

p.position\_name AS Position

FROM Employee e

INNER JOIN Position p ON e.position\_id = p.ID

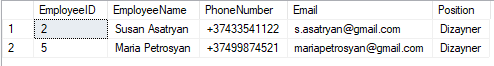
WHERE p.position\_name = @PositionName;

END; END;

RETURN;

END;

SELECT \* FROM dbo.GetEmpByPosition('Dizayner');



3․Ֆունկցիայի կիրառմամաբ դասակարգել ծաղիկները ըստ գների(ցածր, միջին, բարձր)։

CREATE FUNCTION CategorizeFlowersByPrice()

RETURNS TABLE

AS RETURN (

WITH CategorizedFlowers AS (

SELECT F\_id AS FlowerID,

flower\_name AS FlowerName,

flower\_price AS FlowerPrice,

CASE WHEN flower\_price < 500 THEN 'Low'

WHEN flower\_price >= 500 AND flower\_price <= 1000 THEN 'Medium'

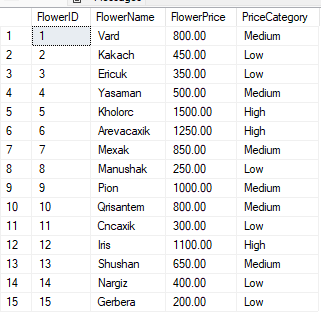
ELSE 'High'

END AS PriceCategory

FROM Flowers )

SELECT \* FROM CategorizedFlowers);

SELECT \* FROM dbo.CategorizeFlowersByPrice();



4․Ֆունկցիայի միջոցով ստանալ աղյուսակ որը տալիս է տեղեկատվություն վաճառքի, աշխատողների և վճարման եղանակի մասին։

CREATE FUNCTION GetEmployeeSalesDetails (@EmployeeID INT)

RETURNS TABLE

AS RETURN (

SELECT S.Sale\_id,

S.flower\_id,

S.sales\_quantity,

S.price,

S.sale\_date,

S.employee\_id,

S.shipping\_id,

S.payment\_method,

CASE WHEN E.gender = 'Male' THEN 'Mr. ' + E.last\_name

WHEN E.gender = 'Female' THEN 'Ms. ' + E.last\_name

ELSE E.first\_name + ' ' + E.last\_name

END AS employee\_name,

CASE WHEN S.payment\_method = 'Cash' THEN 'Paid in cash'

WHEN S.payment\_method = 'Card' THEN 'Paid by card'

ELSE 'Payment method not specified'

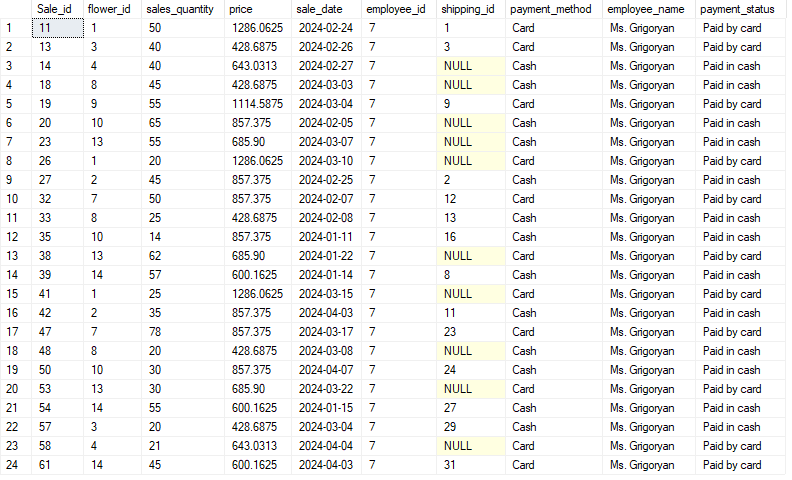
END AS payment\_status

FROM Sales S

INNER JOIN Employee E ON S.employee\_id = E.id

WHERE S.employee\_id = @EmployeeID);

SELECT \* FROM dbo.GetEmployeeSalesDetails(7);



## Տրիգերներ

1․Տրիգերը ապահովում է որ վաճառքի հետ կապ ունենան միայն վաճառողի պաշտոն ունեցող աշխատողները։

CREATE TRIGGER tSalesEmployee

ON Sales

AFTER INSERT, UPDATE

AS BEGIN

IF EXISTS ( SELECT 1

FROM inserted i

JOIN Employee e ON i.employee\_id = e.id

JOIN Position p ON e.position\_id = p.ID

WHERE p.position\_name != 'Vacharox')

BEGIN

IF UPDATE(employee\_id)

BEGIN RAISERROR ('Only employees with the position "Vacharox" can be related to sales. This operation has been aborted due to an update.', 16, 1);

END

ELSE BEGIN

RAISERROR ('Only employees with the position "Vacharox" can be related to sales. This operation has been aborted due to an insert.', 16, 1);

END ROLLBACK TRANSACTION;

END END;

Msg 50000, Level 16, State 1, Procedure tSalesEmployee, Line 19 [Batch Start Line 182]

Only employees with the position "Vacharox" can be related to sales. This operation has been aborted due to an update.

Msg 3609, Level 16, State 1, Line 185

The transaction ended in the trigger. The batch has been aborted.

2․ Տրիգերը հետևում է, որ առաքումները տեղի ունենան ճիշտ վաճառքի օրը։

CREATE TRIGGER ShippingDate

ON Shipping

AFTER INSERT

AS BEGIN

DECLARE @SalesDate DATE, @DeliveryDate DATE, @SaleID INT;

SELECT @SaleID = s.Sale\_id, @DeliveryDate = i.shipping\_date

FROM inserted i

INNER JOIN Sales s ON i.Ship\_id = s.sale\_id;

SELECT @SalesDate = sale\_date

FROM Sales

WHERE Sale\_id = @SaleID;

IF @SalesDate != @DeliveryDate

BEGIN RAISERROR ('Delivery date must match the sales date.', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

Msg 50000, Level 16, State 1, Procedure ShippingDate, Line 24 [Batch Start Line 301]

Delivery date must match the sales date.

Msg 3609, Level 16, State 1, Line 303

The transaction ended in the trigger. The batch has been aborted.

3․ Տրիգերը ապահովում է, որ վաճառքի քանակը չգերազանցի բազայում առկա ծաղիկների քանակին։

CREATE TRIGGER ExcessiveSales

ON Sales

INSTEAD OF INSERT

AS BEGIN

DECLARE @TotalFlowerStock INT;

SELECT @TotalFlowerStock = SUM(flower\_quantity)

FROM Flowers;

DECLARE @TotalSalesQuantity INT;

SELECT @TotalSalesQuantity = SUM(sales\_quantity)

FROM inserted;

IF @TotalSalesQuantity > @TotalFlowerStock

BEGIN RAISERROR ('Transaction aborted: Number of sales exceeds the number of flowers in the database.', 16, 1);

ROLLBACK TRANSACTION;

END ELSE

BEGIN INSERT INTO Sales

SELECT \*FROM inserted;

END END;

Msg 50000, Level 16, State 1, Procedure ExcessiveSales, Line 20 [Batch Start Line 431]

Transaction aborted: Number of sales exceeds the number of flowers in the database.

Msg 3609, Level 16, State 1, Line 432

The transaction ended in the trigger. The batch has been aborted.

4․ Տրիգերը ապահովում է, որ վաճառքից հետո վաճառված ծաղիկների քանակի չափով պակասի բազայում առկա ծաղիկների քանակը։

CREATE TRIGGER SubFlower

ON Sales

AFTER INSERT

AS BEGIN

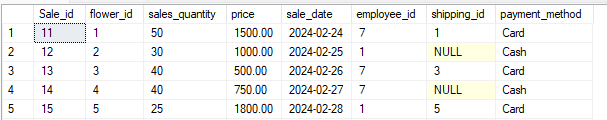
UPDATE Flowers

SET flower\_quantity = flower\_quantity - i.sales\_quantity

FROM Flowers f

INNER JOIN inserted i ON f.F\_id = i.flower\_id;

END;



# 2․4․ Ստեղծել view-եր և դրանց միջոցով կազմակերպել DML հարցումներ

1․ Ստեղծել տեսք որի դաշտերը առաջանում են Sales աղյուսակի որոշակի պայմանի համաձայն։

CREATE VIEW SalesDetailView AS

SELECT s.Sale\_id,

s.flower\_id,

s.sale\_date,

s.employee\_id,

s.sales\_quantity,

s.price,

s.shipping\_id,

s.payment\_method

FROM Sales s

WHERE s.shipping\_id is NULL;

Այդ տեսքի միջոցով կազմակերպել հետևյալ հարցումները։

1․1. Տեսքը լրացնել նոր տվյալներով insert գործողության միջոցով։

INSERT INTO SalesDetailView(Sale\_id, flower\_id,sale\_date, employee\_id, sales\_quantity, price, shipping\_id, payment\_method)

VALUES(100, 2, '2024-04-25', 7, 25, 1000, NULL, 'Card'),

(101, 14, '2024-04-30', 1, 10, 700, NULL, 'Cash' );





1․2․ Եթե վաճառքի միջին քանակը գերազանցում է 150-ը վաճառքի գինը բարձրացնել 20% -ով, հակառակ դեպքում զեղչել 10%-ով։

DECLARE @avg\_sales DECIMAL(10, 2);

SELECT @avg\_sales = AVG(sales\_quantity)

FROM SalesDetailView;

IF @avg\_sales > 150

BEGIN

UPDATE SalesDetailView

SET price = price \* 1.20;

END

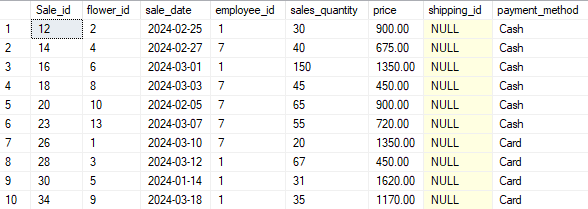
ELSE

BEGIN

UPDATE SalesDetailView

SET price = price \* 0.90;

END



1․3․ Հեռացնել հունվար ամսվա քարտային տարբերակով արված վաճառքների տվյալները։

DELETE

FROM SalesDetailView

WHERE MONTH(sale\_date) = 1;



2․ Ստեղծել տեսք որի դաշտերը առաջանում են Employee, Flowers և Sales աղյուսակների միավորումից և բավարարում են տրված պայմանին։

CREATE VIEW SalesEmpFLView AS

SELECT s.Sale\_id,

s.sale\_date,

s.sales\_quantity,

s.price,

s.payment\_method,

f.flower\_name,

f.flower\_price,

e.first\_name AS employee\_first\_name,

e.last\_name AS employee\_last\_name

FROM Sales s

INNER JOIN Flowers f ON s.flower\_id = f.F\_id

INNER JOIN Employee e ON s.employee\_id = e.id

WHERE flower\_price>=500 And CAST(RIGHT(phone\_number, 1) AS INT) % 2 = 1;

Այդ տեսքի միջոցով կազմակերպել հետևյալ հարցումը.

2․1․ Եթե վաճառքի քանակը գերազանցում է 50-ը և վաճառքը իրականացվել է կանխիկ տարբերակով ծաղիկների գինը զեղչել 5%-ով, հակառակ դեպքում քարտային վճարման դեպքում ծաղիկների գինը բարձրացնել 10%-ով։

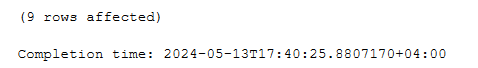
UPDATE SalesEmpFLView

SET flower\_price = CASE WHEN payment\_method = 'Card' THEN flower\_price \* 1.1

WHEN payment\_method = 'Cash' AND sales\_quantity > 50 THEN flower\_price \* 0.95

ELSE flower\_price

END;





# Եզրակացություն

Եզրափակելով, այս աշխատանքը տվեց մեծ փորձառություն ծաղիկների տվյալների բազայի կայուն համակարգի ստեղծման գործում: Բազայում պահվում է տեղեկատվություն ծաղիկների,աշխատողների, առաքումների, վաճառքի և մատակարարաների մասին։ Աշխատանքի շրջանակներում ստեղծվեցին և մշակվեցին զանազան օբյեկտներ և հայցեր։ Բազայի աղյուսակների ստեղծման, սահմանափակումների մանրակրկիտ կիրառման, պահպանող մոդուլների կիրառման, ինչպիսիք են պրոցեդուրա, ֆունկցիա և տրիգեր մշակվեց համապարփակ լուծում՝ հարմարեցված ծաղիկների վաճառքի բիզնեսի հատուկ տրամաբանության: Տեսքերի կիրառումը հնարավորություն է տալիս խուսափել բարդ ներկառուցված հայցեր ստեղծելուց։ Այս կարևոր գործիքների օգտագործումը բարելավում է տվյալների բազայի սխեմայի ընդհանուր կառուցվածքը, կազմակերպումը և ճկունությունը։

# Գրականություն

● Տվյալների հենքերի կառավարման համակարգեր (SQL): Դասագիրք / Ա.Շ. Սիմոնյան; ՀԱՊՀ.-Եր.: Ճարտարագետ, 2017. - 336 էջ:

● <https://www.w3schools.com/sql/default.asp>

● https://learn.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver16